

## **REMARKS**

Reconsideration and allowance are respectfully requested. By this Amendment, Claims 8, 10, 13, 16, 17, 24, and 25 haven been amended. New claims 27-30 having been added. Therefore, Claims 1-30 are pending.

### **I. Claim Objections**

Applicants have rewritten Claim 16 in independent form including the limitations of Claim 13. As such, the Examiner's objection to Claim 16 is overcome and Claim 16 is allowable. Claims 8 and 24 included similar subject matter as Claim 16 and have been rewritten in independent form including the limitations of the base claim and any intervening claim. Therefore, Claim 8, Claim 9, Claim 24, and Claim 25 are allowable.

### **II. Claim Rejections – 35 U.S.C. § 103**

The Examiner rejected Claims 1-15 and 17-26 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5,963,966 issued to Mitchell et al. (hereinafter "Mitchell"). Applicants traverse the rejections because the reference does not teach or suggest every limitation of the rejected claims. Further, the limitations of the rejected claims are not obvious in view of Mitchell.

#### **A. Claim 1**

Independent Claim 1 recites:

An output engine operable to convert a file from a first format to a second format, the output engine comprising:

a decomposer operable to be called by a calling application and to receive a file and a desired file format from the calling application, the decomposer operable to decompose the file into a component architecture; and

a writer operable to call the decomposer multiple times to retrieve the component architecture of the file and generate a new version of the file in the second format.

Mitchell does not teach or suggest “a writer operable to call the decomposer multiple times,” as recited in Claim 1 (emphasis added).

Mitchell discloses an optical character recognition (hereinafter “OCR”) document conversion system “which translates document images into ASCII and graphics components” (Mitchell, col. 1, lines 57-58). The OCR document conversion system disclosed performs “page decomposition, where each page is segmented into graphic and text regions” (Mitchell, col. 1, lines 59-61). As shown in FIG. 1 of Mitchell, page decomposition results in one or more graphical regions and one or more textual regions that are used by another application or system for performing additional operations. For example, operations performed on the one or more graphical regions can include “extracting text from within graphics” and performing “raster-to-vector conversion” (Mitchell, col. 1, lines 62-64). Operations performed on the one or more textual regions can include computing OCR for the text regions, as well as [for] text located within the graphical regions” (Mitchell, col. 1, lines 64-65).

After performing the above operations of the one or more graphical regions and the one or more textual regions, Mitchell discloses “[a]n integration step [that] combines the results of the graphical and textual processes into a final electronic format” (Mitchell, col. 1, lines 65-67).

As indicated by the Examiner, Mitchell does not disclose a writer calling a decomposer multiple times. The Examiner asserts that Mitchell discloses an integrator that integrates graphic and text file and that it would be obvious that [the integrator] call a decomposer multiple times. With due respect, Applicants disagree. FIG. 1 of Mitchell illustrates a page decomposition process that outputs one or more graphic regions and one or more text regions. As noted by the Examiner, Mitchell does not explicitly disclose calling the decomposer multiple times. And, in contrast to the Examiner’s interpretation, it appears that the integrator disclosed by Mitchell receives the one or more graphic regions and the one or more text regions from intermediary programs and does not call the page decomposition process at all.

FIG. 1 of Mitchell discloses an integrator (i.e., the integration block) that obtains one or more graphic regions that are first processed by a commercial product performing raster-to-

vector conversion in order to extract text from graphics and perform text recognition. The integrator also obtains one or more text regions that are first processed by a commercial product performing OCR and text recognition. If the integrator were to call a process or product, the integrator would call the commercial product processing the graphic regions once and would call the commercial product processing the text regions once. Mitchell, therefore, does not disclose the integrator calling the page decomposition process even once and does not disclose the integrator calling any particular process or product (e.g., the commercial products) more than once.

The Examiner further asserted that since Mitchell discloses an integrator integrating graphic and text files it would “be obvious for a person with ordinary skill in the art at the time the invention was made to incorporate [a] Writer calling multiple times in the method disclosed by Mitchell because it provides multitasking that is always desired.” Calling an application or a product multiple times does not automatically provide multitasking. Multitasking occurs when two event occur simultaneously or in parallel. Performing multiple calls is simply making a plurality of calls to or repeatedly calling the decomposer. For that reason, the Examiner cannot assert that a writer calling a decomposer multiple times, as recited in Claim 1, provides multitasking. In addition, Applicants disagree that multitasking is always desired. For example, so-called multitasking in operating systems such as Windows is often no more than a coordinated switching of CPU and memory resources from one program to another, but at any one time only one program is executed. Furthermore, when multitasking of numerous programs is attempted or if multitasking is attempted on a system without sufficient CPU, memory, or bus capabilities, computer operation either slows to a crawl or fails. These problems are readily apparent to anyone who has run, for example, anti-virus and other utility software while at the same time trying to run, a second program, for example, Microsoft Word. What is often observed is that operation of the Word software slows to an unacceptable level while the anti-virus or utility software attempts to execute simultaneously. Multitasking negative side-effects, such as slow operation, can result from increasing the amount of applications held in memory and therefore cause memory management issues, such as thrashing and page faults. Similar issues can also be

observed when segmenting a single program into one or more threads and allowing the threads to process simultaneously.

In summary, the Examiner's reliance on multitasking to provide motivation to modify the teachings of the reference fails. First, the Applicants are focused on multiple or repeated calls not multitasking. Second, multitasking is not always desired. The effectiveness of multitasking depends on numerous factors such as those noted above and Applicants disagree that there is any maxim to the contrary.

For at least the reasons set forth above, Independent Claim 1 and dependent Claims 2-7, which depend on Claim 1, are allowable. Dependent Claims 2-7 also contain additional features that are not taught or suggested in Mitchell. For example, Mitchell does not disclose an output engine with two writers as recited in Claim 2, where each of the two writers is configured to call the decomposer multiple times. Mitchell does not disclose a decomposer with a job processor. Mitchell teaches page decomposition and then subsequent processing of the decomposed material not a job processor within the decomposer. As such, the specific details about the job processor claimed in Claims 4 – 7 are not shown. For example, associating data with each page as required in Claim 4 is not shown in Mitchell as being handled by a job processor included in the decomposer.

#### **B. Claims 10**

Amended independent Claim 10 recites:

A method of converting an input file from a first format to a second format, the method comprising:

delivering a desired file format and the input file to a decomposer;

decomposing the input file into a component architecture in the decomposer, the component architecture including properties of objects included in the input file;

making the component architecture available to a writer; and

generating a new version of the input file in the second format by calling the decomposer multiple times from a writer to obtain the properties of the objects included in the input file.

Mitchell does not teach or suggest “decomposing the input file into a component architecture in the decomposer, the component architecture including properties of objects included in the input file” and “generating a new version of the input file in the second format by calling the decomposer multiple times from a writer to obtain the properties of the objects included in the input file,” as recited in amended Claim 10.

As noted above with respect to Claim 1, Mitchell does not teach or suggest, “calling a decomposer multiple times from a writer.” However, to further clarify calling a decomposer multiple times from a writer as described in an embodiment of the present application, Claim 10 has been amended to include, among other elements, “decomposing the input file into a component architecture in the decomposer, the component architecture including properties of objects included in the input file” and “generating a new version of the input file in the second format by calling the decomposer multiple times from a writer to obtain the properties of the objects included in the input file” (amendments underlined).

As described above, Mitchell discloses an OCR document conversion system that performs page decomposition that segments each page into graphic and text regions and integration that combines the graphic and text regions into a final electronic format.

In contrast, the present application discloses a decomposer that “loads each page of a form or document within the input file 92, one page at a time. The decomposer 90 also associates data with each page of the input file...The decomposer 90 also drives each page to an appropriate writer...[T]he decomposer 90 decomposes the properties of the file being converted to a new format and makes those decomposed properties available to the writers 100” (page 8, lines 7-12). Therefore, in some embodiments of the invention, the decomposer decomposes pages of an input file into properties, drives the pages to a writer, and allows the writer to call or request the decomposed properties of the input file. As noted above, Mitchell discloses segmenting a page into graphic and text regions and does not disclose “decomposing the input

file into a component architecture in the decomposer, the component architecture including properties of objects included in the input file” and “generating a new version of the input file in the second format by calling the decomposer multiple times from a writer to obtain the properties of the objects included in the input file,” as recited in amended Claim 10.

For the reasons set out above, independent Claim 10 and dependent Claims 11-12 are allowable. Dependent claims 11-12 also contain additional features that are not taught or suggested in Mitchell.

### **C. Claims 13 and 17**

For the reasons set out above with respect to Claim 1 and Claim 10, independent Claims 13 and 17 and dependent claims 14-15, 18-23, and 26 are allowable. Dependent claims 14-15, 18-23, and 28 also contain additional features that are not taught or suggested in Mitchell. For example, Mitchell does not disclose an output engine with two writers, where each writer is operable to call the decomposer multiple times as required in Claim 18. Mitchell does not disclose a decomposer with a job processor and the specific features of the job processor as set out in Claims 18 – 22. Furthermore, Mitchell does disclose a writer with the features set out in Claims 23 – 26.

New Claims 27 – 29 are directed to a method of converting an input file to a second format. Among other things, the method requires delivering the desired file format and the input file to the decomposer, decomposing the input file into a component architecture, generating a data file representing an aspect of the document, generating an object model for the data file, making the object model available to a writer, and calling the decomposer multiple times from a writer. New Claim 28 requires, among other things, delivering a writer identifier to the decomposer and selecting a writer based on the writer identifier. New Claim 29 requires, among other things, loading each page of the input file and associating data with each page in a job processor. New Claim 30 is directed to an output engine that, among other things, includes a decomposer and a plurality of writers where a set of parameters is used to select the writers and each writer is configured to call the decomposer multiple times. These claims are allowable for

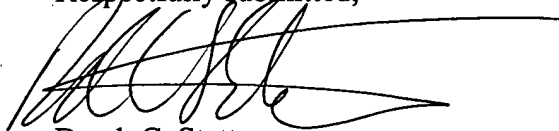
the reasons noted above and include additional patentable features such as, for example, selection of a writer based on parameters.

Support for Claims 27 – 30 can be found in the specification at page 7, line 24 – page 8, line 18 and page 10, lines 11 – 21, among other places.

### **III. Conclusion**

All objections and rejections have been addressed. It is respectfully submitted that the present application is now in condition for allowance, and a notice to that effect is earnestly solicited. Should there be any questions or concerns regarding this application, the Examiner is invited to contact the undersigned at the below-listed telephone number.

Respectfully submitted,

A handwritten signature in black ink, appearing to read 'Derek C. Stettner', with a long horizontal line extending to the right.

Derek C. Stettner  
Reg. No. 37,945

Docket No.: 200887-9002-00  
Michael Best & Friedrich LLP  
401 North Michigan Avenue  
Suite 1900  
Chicago, Illinois 60611  
312.222.0800

S:\client\200887\9002\A1235797.3